Preconditioning issues in the numerical solution of nonlinear equations and nonlinear least squares *

Stefania Bellavia† Margherita Porcelli ‡

March 31, 2014

Abstract

Second order methods for optimization call for the solution of sequences of linear systems. In this survey we will discuss several issues related to the preconditioning of such sequences. Covered topics include both techniques for building updates of factorized preconditioners and quasi-Newton approaches. Sequences of unsymmetric linear systems arising in Newton-Krylov methods will be considered as well as symmetric positive definite sequences arising in the solution of nonlinear least-squares by Truncated Gauss-Newton methods.

1 Introduction

We consider sequences of linear systems of the form:

$$A_k s = b_k \quad k = 0, 1, \ldots \tag{1}$$

where $A_k \in \mathbb{R}^{n\times n}$ is large, sparse, nonsingular and b_k is the corresponding right-hand side. The matrix A_k and the vector b_k change from one system to the next and they are not available simultaneously. Newton-type methods for optimization problems require the solution of linear systems of the form (1), whose characteristics of the matrices A_k depend on the class of problems considered and on the specific method used. Here, we will focus on sequences arising in optimization methods for nonlinear systems and nonlinear least-squares problems and assume to use a Krylov subspace method to solve the linear systems.

It is well-known that a clever solution of the linear algebra phase required in most nonlinear optimization methods is crucial for their practical implementation [20] and it is widely recognized that preconditioning is a critical ingredient for an efficient iterative solution of linear systems. There is a continuum of choices for preconditioning of the sequence $\{A_k\}$ which range from freezing the preconditioner computed for the first matrix of the sequence to recomputing a preconditioner from scratch for each matrix. Recomputing the preconditioner for each matrix of the sequence may be too expensive and pointless accurate. On the other hand, freezing the preconditioner may yield a low convergence or a failure of the Krylov solver. A compromise

†Dipartimento di Ingegneria Industriale, Università di Firenze, viale G.B. Morgagni 40, 50134 Firenze, Italia, stefania.bellavia@unifi.it

‡Dipartimento di Matematica, Università di Bologna, Piazza di Porta S. Donato 5, 40127 Bologna, Italia, margherita.porcelli@unibo.it

*This work was supported by *National Group of Computing Science (GNCS-INDAM)*: GNCS 2013 Projects "Metodi numerici e software per l'ottimizzazione su larga scala con applicazioni all'Image Processing" and "Strategie risolutive per sistemi lineari tipo KKT con uso di informazioni strutturali", GNCS 2013-2014 Project "Identificazione automatica dei parametri algoritmici ottimali".

between these two approaches is given by updating techniques where efficient preconditioners for the current system are derived from previous systems of the sequence in a cheap way. This way, the expensive computation of a new preconditioner is avoided and some of the computational effort is shared among the linear systems of the sequence. The expected performance of an updated preconditioner in terms of linear iterations are in between those of the frozen and the recomputed preconditioners, while an overall saving in terms of computational time is expected. We note that updated preconditioners may deteriorate in practice along the sequence and that therefore the definition of suitable strategies for adaptively refresh the preconditioner when needed represents a decisive issue.

In this semi-survey we focus on updating procedures giving rise to implicit and/or explicit preconditioners. A preconditioner is implicit if its application, within each step of the Krylov method, requires the solution of a linear system. For example, preconditioners based on incomplete factorization of the matrix A_k are of implicit kind. On the other hand, an explicit preconditioner provides an approximation of the inverse of A_k and the preconditioning operation reduces to computing matrix-vector products. Approximate inverse preconditioners fall in this class [10, 11]. Both quasi-Newton-type updates [12, 13, 34, 35, 37] and fully-algebraic procedures like those in [2, 8, 15, 24, 25] will be covered. Methods falling in the first class update the preconditioners by matrices of small rank, using quasi-Newton formula. The obtained preconditioners are of explicit kind. Methods in the second class build implicit and/or explicit preconditioners starting from an incomplete factorization of a specific matrix of the sequence or of its inverse. Then, preconditioners for the matrices of the sequences are obtained by cheap updates of such a factorization. We will also consider situation where the matrix A_k is not available, but matrix-vector products with A_k can be computed or approximated by operators. In other words matrix-free approaches are also discussed.

Other approaches are possible, we mention preconditioners based on recycled Krylov information [27, 33, 41] and Incremental ILU preconditioners [18]. We do not cover these approaches in this semi-survey.

The paper is organized as follows. In Section 2 we describe classical optimization algorithms for nonlinear systems and nonlinear least-squares problems which give rise to the sequences we are interested in. Section 3 is devoted to the description of nearly matrix-free preconditioning strategy. In Sections 4 and 5 we focus on preconditioners based on updating a given incomplete factorizations and based on quasi-Newton-type updates, respectively. Finally, Sections 6 and Section 7 concern the analysis of the practical implementation and of the numerical behaviour of the described strategies.

Notations

Throughout the paper, the subscript k will denote an iteration counter and for any function h, h_k will be the shorthand for $h(x_k)$. Unless explicitly stated, $\|\cdot\|$ denotes an arbitrary vector norm and induced matrix norm. The entries of a matrix $A \in \mathbb{R}^{n\times n}$ are denoted either as a_{ij} or $(A)_{ij}$. Given a nonnegative integer γ, $[A]_\gamma \in \mathbb{R}^{n\times n}$ indicates the band matrix obtained extracting from A the main diagonal and γ upper and lower diagonals. If $\gamma = 0$, $[A]_0$ is the diagonal matrix formed from the elements of the diagonal of A. The off-diagonal part $A - [A]_0$ of A is denoted by the symbol $\mathit{off}(A)$. If A is lower (upper) triangular, $[A]_\gamma$ is obtained extracting from A the main diagonal and γ lower (upper) diagonals. Similarly, $\mathit{off}(A)$ is formed by the lower (upper) extra diagonals of A. We borrow notations used by MATLAB in linear algebra: *diag, tril, triu*. Given a vector v the notation $(v)_i$ denotes the i-th component of v, when clear from the context the brackets are dropped. Finally, P_k and B_k denote, respectively implicit and explicit

preconditioners for A_k, that is $P_k \simeq A_k$, while $B_k \simeq A_k^{-1}$.

2 Applications

In this section we briefly describe optimization methods for nonlinear systems and nonlinear least-squares problems. We focus on the linear algebra phase enlightening that sequences of linear systems of the form (1) arise and describing the special structure of the matrices involved. Then, we turn our attention to the following two classes of problems:

- Nonlinear systems:
$$\Theta(x) = 0, \tag{2}$$
where $\Theta : \mathbb{R}^n \to \mathbb{R}^n$ is continuously differentiable.
- Nonlinear least-squares problems
$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2}\|F(x)\|_2^2, \tag{3}$$
where $F : \mathbb{R}^n \to \mathbb{R}^m$ is continuously differentiable.

Let Θ' and F' be the Jacobian matrices of Θ and F, respectively.

2.1 Sequence arising in Newton-Krylov methods

Newton-Krylov methods [6, 17, 21] applied to the nonlinear system (2), require the solution of the following sequence of linear systems:

$$\Theta'_k s = -\Theta_k, \qquad k = 0, 1, \ldots \tag{4}$$

where x_k is the current iteration. Such linear systems are approximately solved by a Krylov method. More precisely, an Inexact-Newton step s_k^I satisfying

$$\|\Theta'_k s + \Theta_k\| \leq \eta_k \|\Theta_k\| \qquad \eta_k \in (0, 1) \tag{5}$$

is computed. In (5) η_k is the so-called forcing term. It is used to control the accuracy in the solution of the system (4) and its choice influences the convergence rate of the Newton-Krylov procedure. These methods are often embedded in suitable globalization strategies [1, 5, 26].

Then, a sequence of the form (1) has to be solved, with

$$A_k \equiv \Theta'_k, \quad b_k \equiv -\Theta_k.$$

Generally, the Jacobians Θ'_k are sparse unsymmetric matrices but there are also a number of applications such as e.g., unconstrained optimization, or discretization of nonlinear PDEs with Picard linearization, where Θ'_k are symmetric.

By continuity, matrix Θ'_k varies slowly from one step k to the next whenever the iterates x_k and x_{k+1} are sufficiently close; e.g this is the case when the iterates are close enough to a solution. The action of the Jacobians times a vector v required by the Krylov solver is usually provided by an operator or is approximated by finite-differences, i.e.

$$\Theta'_k v \simeq \frac{\Theta(x_k + \epsilon v) - \Theta_k}{\epsilon \|v\|} \tag{6}$$

where ϵ is a proper chosen positive constant. This is computed at cost of one Θ-evaluation. We refer to [16] for details on the approximation of Jacobian-vector product by finite-differences and on the convergence analysis for variants of Newton-Krylov methods that employ such approximation.

2.2 Truncated Gauss-Newton methods for nonlinear least-squares

Consider now the solution of the nonlinear least-squares problem (3) by a Truncated Gauss-Newton method [28, 29, 42]. At each iteration k, a search direction s_k^I is computed by approximately minimizing the quadratic model

$$m_k(s) = \frac{1}{2}\|F_k + F_k' s\|_2^2$$

over $\mathbb{R}^n$ and then a moving step to update the current iterate x_k is obtained by imposing a suitable sufficient decrease condition [28, 42].

For the computation of the step s_k^I, a CG-like method [30, 40] is applied to the normal equations

$$F_k'^T F_k' s = -F_k'^T F_k, \tag{7}$$

until a prescribed reduction of the value of ∇m_k is produced, i.e.

$$\|\nabla m_k(s)\| \leq \eta_k \|\nabla m_k(0)\|, \tag{8}$$

where $\eta_k \in [0, 1)$ is the forcing term.

Summarizing, in a Truncated Gauss-Newton framework a sequence of the form (1) is generated with

$$A_k \equiv F_k'^T F_k', \quad b_k \equiv -F_k'^T F_k.$$

We remark that the matrices A_k are symmetric positive semidefinite for general nonlinear least squares problems but that they are positive definite whenever $m \geq n$ and F_k' is full rank. Note that if CG is initialized with the null vector, CG terminates in a finite number of iterations computing the minimum norm step (the Newton step), see [30]. We further underline that CG-like methods require the action of both $F_k'^T$ and F_k'. The action of F_k' on a vector v can be approximated by finite differences, employing formula (6) replacing Θ by F and Θ' by F'.

3 Nearly matrix-free preconditioning

Krylov methods applied to linear systems (1) can be implemented in a matrix-free manner, i.e. without requiring the computation of the matrix A_k, provided that an operator performing the product of A_k times a vector is available. Then, in principle, Newton-Krylov methods for nonlinear systems as well as Truncated Gauss-Newton methods for unconstrained minimization can be implemented in a matrix-free manner whenever such operator is available. As already noted, for sequences arising in the Newton-Krylov framework a possible choice is to approximate the action of the Jacobian on a vector via the finite differences operator.

On the other hand, when an algebraic preconditioner is used, the full matrix A_k is needed and if it is not available it has to be computed or approximated via the operator performing the product of A_k times a vector. This task is usually expensive. For example, in the Newton-Krylov context, when the Jacobian is not explicitly available, the most relevant part of the computational time is devoted to the approximation of the Jacobian by finite-differences. Therefore, in the optimization community there has been interest in updating strategy that can be implemented in a *nearly matrix-free* manner, that is close to true matrix-free settings. Specifically, nearly matrix-free preconditioning has the following properties: a few full matrices are formed; for preconditioning most systems of the sequence, matrices that are reduced in complexity with respect to the full A_k's are required; matrix-vector product approximations by finite differences can be used; see [25, 32].

Note that a preconditioning strategy that requires only the computation of selected elements of A_k, can be considered nearly-matrix free whenever the function that performs the matrix-vector product is separable. More precisely, let $\mathcal{F}$ be the function that, evaluated at $v \in \mathbb{R}^n$, provides the product of A_k times v. $\mathcal{F}$ is said separable if its evaluation can be easily separated in the evaluation of its function components, i.e. computing one component of $\mathcal{F}$ costs about an n-th part of the full function evaluation [25]. When $\mathcal{F}$ is the finite-differences operator, $\mathcal{F}$ is separable whenever the nonlinear function itself is separable.

In the situation where $\mathcal{F}$ is separable and only selected entries of the current matrix A_k are required to build the preconditioner, the preconditioning strategy may be considered nearly matrix-free as the evaluation of selected elements of A_k can be considerably less costly than forming the whole matrix. In fact, note that the entry $(A_k)_{ij}$ is given by the i-th component of the vector $A_k e_j$, where e_j is the j-th vector of the canonical base. Then,

$$(A_k)_{ij} = \mathcal{F}_i(e_j)$$

and the cost of evaluating a selected entry of A_k corresponds approximately to the n-th part of the cost of performing one matrix-vector product. This implies that evaluating the diagonal of A_k costs about as performing one matrix-vector product.

4 Updating of preconditioner's factorization

The approaches proposed in [2, 8, 15, 24, 25] focus on updating the factorization of a preconditioner for a specific matrix A_s of the sequence with the aim of building preconditioners for the subsequent matrices. These processes give rise to factorized preconditioners that are products of matrices easy to invert. Clearly, in all the proposed procedure the updating is obtained at a low computational cost, significantly lower than the cost of computing a new preconditioner from scratch. In what follows, we will refer to A_s as the *seed* matrix, to P_s and B_s as the seed preconditioner in the implicit and explicit form, respectively. Furthermore, it is assumed that the matrix A_s is available and an incomplete factorization process of A_s and A_s^{-1} can be carried out without breakdowns.

The preconditioner updates in [2, 8, 15, 24, 25] are inspired by the attempt to cheaply approximate the *ideal* update to an existing preconditioner [24]. Let

$$P_s = LDU, \tag{9}$$

be an incomplete ILU factorization of A_s where D is diagonal and L and U are lower and upper triangular matrices, respectively, with unit main diagonal [44].

Then, $A_s \approx LDU$ and using the equality $A_k = A_s + (A_k - A_s)$ we get

$$A_k \approx L(D + L^{-1}(A_k - A_s)U^{-1})U. \tag{10}$$

Therefore, the matrix

$$P_k^I = L(D + L^{-1}(A_k - A_s)U^{-1})U, \tag{11}$$

represents the ideal updated preconditioner in the sense that, for any matrix norm $\|\cdot\|$, its accuracy $\|A_k - P_k^I\|$ for A_k is equal to the accuracy $\|A_s - P_s\|$ [24].

It is evident that the ideal update is not suitable for practical use. In general, the matrix $L^{-1}(A_k - A_s)U^{-1}$ is expensive to build; moreover P_k^I is dense and its factorization is impractical. Both structured and unstructured updates have been proposed where approximations of the factors of P_k^I are employed. The term *structured* refers to the fact that the approximations used

have a special structure. Unstructured updatings based on Gauss-Jordan transformations have been proposed in [24]. Here, we will deal with updating preconditioner strategies which use structured approximation of the factors of P_k and differ in the way the term $L^{-1}(A_k - A_s)U^{-1}$ is approximated. These strategies are nearly-matrix free whenever the function $\mathcal{F}$ performing matrix-vector products is separable.

In particular, the inverse of matrices L and/or U are approximated by simpler matrices obtained sparsifying them dropping some of their elements or retaining only their main diagonal and/or a few upper and lower diagonals. This is motivated by large class of matrices where L^{-1} and U^{-1} contain many entries which are small in magnitude. Classes of matrices with decaying inverses, i.e. with entries of the inverse that tend to zero away from the main diagonal, were detected and analyzed in several papers: banded symmetric positive definite and indefinite matrices [22, 36]; nonsymmetric block tridiagonal banded matrices [38]; matrices of the form $f(A)$ where A is symmetric and banded and f is analytic [9]. Typically, the rate of decay of the entries of A^{-1} is fast if A is diagonally dominant. As an example, we plot in Figure 1 the sparsity pattern (on the left) and the wireframe mesh (on the right) of the 2D Laplacian matrix and its inverse, respectively. Note that 2D Laplacian matrix is banded (tridiagonal) and diagonally dominant.

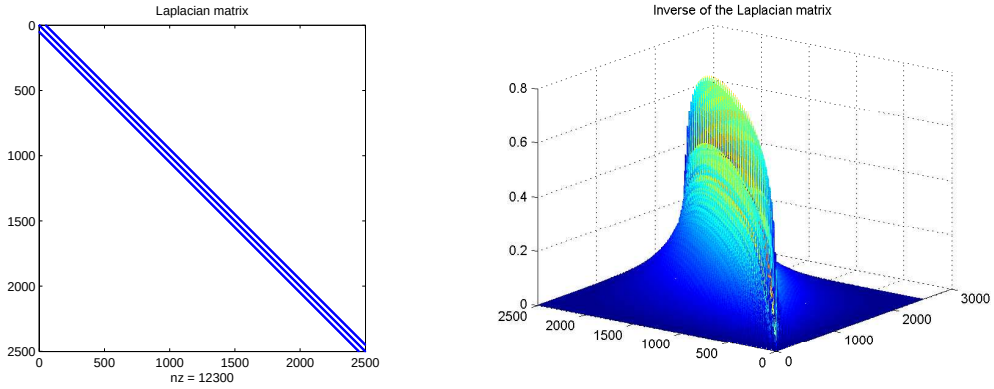

Figure 1: The 2D Laplacian matrix A: sparsity pattern of A (on the left) and wireframe mesh of its inverse A^{-1} (on the right) ($n = 2500$).

4.1 Updates based on triangular approximation of the current matrix

A first approach in the definition of nearly matrix-free updating procedures is due to Duintjer Tebbens and Tůma in [24, 25].

The practical updating techniques proposed in [24] are based on the implicit preconditioner P_s given in (9). In order to obtain a factorized preconditioner for A_k, in (11) the matrix $A_k - A_s$ is replaced by a nonsingular and easily to be inverted approximation and either L^{-1} or U^{-1} are approximated by the identity matrix. The approximation consists of two steps. First, the ideal update (10) is approximated by

$$A_k \approx L(\, D + (A_k - A_s)U^{-1}\,)U = L(\, DU + (A_k - A_s)\,), \tag{12}$$

or

$$A_k \approx L(D + L^{-1}(A_k - A_s))U = (LD + (A_k - A_s))U. \tag{13}$$

Note that, in the first case L it is assumed to be close to the identity matrix and L^{-1} is neglected in (10), while, in the second case U^{-1} is discarded relying on the assumption that U is close to the identity matrix.

Several options have been proposed in [24] to replace $DU + (A_k - A_s)$ or $LD + (A_k - A_s)$ by a nonsingular and easily invertible approximation. One option is to approximate $A_k - A_s$ in (12) or (13) by its upper/lower triangular part which gives rise to

$$P_k^U = L(\, DU + triu(A_k - A_s)), \tag{14}$$

$$P_k^L = (LD + tril(A_k - A_s))U, \tag{15}$$

respectively. This way the preconditioner is available in a factorized form and is obtained employing information on A_k reduced in complexity with respect to the full matrix. The cost for applying this preconditioning is the solution of two triangular systems. Clearly, P_k^U and P_k^L are expected to be accurate when the factor U and L are close to the identity matrix and $tril(A_k - A_s)$ or $triu(A_k - A_s)$ are useful approximation of $A_k - A_s$, that is one triangular part of $A_k - A_s$ contains more relevant information than the other part. A typical situation of this kind arises when matrices come from upwind/downwind discretization schemes [24] and in compressible flow problems [15].

In order to use the above triangular updates the upper or lower triangular part of A_k is needed. Interestingly, the explicit computation of such triangular parts may be avoided and nearly matrix-free implementations of this approach are possible. Two possible approaches have been proposed in [25]. The first approach gives rise to a partial matrix estimation problem whose solution produces an approximation of the required triangular part. This proposal has been enhanced in [43] where a two-sided bicolouring method is used. The second approach, that we will now briefly describe, avoids matrix estimation and assumes separability of the function $\mathcal{F}$ performing matrix-vector products. Consider the update (15). The application of P_k^L requires the solution of the lower triangular systems

$$(E + tril(A_k))z = v$$

where the matrix $E = LD - tril(A_s)$ is known. Then, the vector z is computed as follows:

$$z_i = \frac{v_i - \sum_{j<i}(E)_{ij} z_j - \sum_{j<i}(A_k)_{ij} z_j}{e_{ii} + (A_k)_{ii}}, \qquad i = 1, \ldots, n.$$

The term $\sum_{j<i}(A_k)_{ij} z_j$ required in the above computation is the i-th component of the vector $A_k \bar{z}$, with $\bar{z} = (z_i, z_2, \ldots, z_{i-1}, 0, \ldots, 0)^T$ and therefore

$$\sum_{j<i}(A_k)_{ij} z_j = \mathcal{F}_i(\bar{z}) \quad i = 1, \ldots, n. \tag{16}$$

Then, the application of the preconditioner in a matrix-free manner can be performed as follows. First, before solving the linear system, the diagonal of A_k is computed. Then, each application of the preconditioner within the Krylov solver is performed exploiting (16). This calls for the evaluation of n components of the function $\mathcal{F}$ and if $\mathcal{F}$ is separable, it is accomplished at the cost of one $\mathcal{F}$-evaluations. Therefore, this updating procedure can be performed in a nearly matrix-free manner requiring one $\mathcal{F}$-evaluation for computing the diagonal of A_k and one $\mathcal{F}$-evaluation at each iteration of the Krylov solver, in order to apply the preconditioner. In the solution of nonlinear systems (4), when $\mathcal{F}$ is the finite difference operator and Θ is separable, this cost is approximately the cost of one Θ-evaluations for each iteration of the Krylov solver.

4.2 Updates based on banded approximation of the current matrix

In this subsection we review the updating strategies proposed in [2, 8] which differ from those discussed in the previous section. In fact, the strategies in [24, 25] are based on the observation that P_k^I requires the knowledge of L^{-1} and U^{-1} and avoid the computation of such inverses approximating one of them with the identity matrix. On the other hand, the idea in [2, 8] is to retain more information on the inverse of L or U and to exploit information from both the lower and upper triangular parts of the current matrix instead of discarding either the lower or the upper part as in the approach previously described.

The techniques proposed in [8] attempt to approximate P_k^I by extracting banded parts of the matrices appearing in it. In particular, the matrix $A_k - A_s$ in P_k^I is replaced by the band matrix $[A_k - A_s]_\beta$. Let us consider, first, the case $\beta > 0$. In this case, band approximate inverses of L and U are built in order to replace L^{-1} and U^{-1} in (10). An algorithm for computing exactly the matrices $[L^{-1}]_\gamma$ and $[U^{-1}]_\gamma$ without the complete inversion of L and U has been proposed in [8] and we refer to [8] for its description. Here, we underline that each row of $[U^{-1}]_\gamma$ (column of $[L^{-1}]_\gamma$) can be computed independently from the others and the computation can be carried out in parallel. Once $[L^{-1}]_\gamma$ and $[U^{-1}]_\gamma$ have been calculated, the seed preconditioner LDU is updated as follows:

$$P_k^B = L \left[D + [L^{-1}]_\gamma [A_k - A_s]_\delta [U^{-1}]_\gamma \right]_\beta U \tag{17}$$

giving rise to an implicit preconditioner for A_k. In case the value $\delta = 0$ is used, i.e. only the diagonal of the difference $A_k - A_s$ is retained, the ideal preconditioner (10) simplifies to

$$A_k \approx LDU + \Sigma_k,$$

where

$$\Sigma_k = [A_k - A_s]_0$$

and the updating procedure is carried out without involving L^{-1} and U^{-1}.

In this situation, in order to obtain a factorized preconditioner and make $LDU + \Sigma_k$ of practical use, in [8] the following strategy is proposed to compute a cheap approximate factorization of the form

$$P_k^D = L_k D_k U_k \approx LDU + \Sigma_k. \tag{18}$$

Given the seed preconditioner $P_s = LDU$ the factors L_k, D_k, U_k in (18) are obtained updating the seed preconditioner factorization as follows

$$D_k = D + \Sigma_k, \tag{19}$$

$$S_k = diag((S_k)_{11}, \ldots, (S_k)_{nn}), \qquad (S_k)_{ii} = \frac{|(D_k)_{ii}|}{|(D_k)_{ii}| + |(\Sigma_k)_{ii}|}, \ i = 1, \ldots, n, \tag{20}$$

$$diag(L_k) = diag(U_k) = (1, \ldots, 1)^T,$$
$$off(L_k) = off(L) S_k, \qquad off(U_k) = S_k\, off(U). \tag{21}$$

The previous definition is motivated by the desire for having preconditioners that can be applied at the same cost as the seed preconditioner and that mimic in some sense the behavior of the corresponding matrices $LDU + \Sigma_k$. In fact, by the form of the diagonal scaling matrix S_k, the off diagonal parts of L_k and U_k decrease in absolute value as the entries of Σ_k increase, i.e. when the diagonal of $LDU + \Sigma_k$ tends to dominate over the remaining entries. On the other hand, when the entries of Σ_k are small then $LDU + \Sigma_k$ is close to LDU, S_k is close to the unit matrix, and the factors L_k and U_k are close to L and U respectively. Moreover, the sparsity pattern of the factors of P_s is preserved and the cost to form P_k^D is low since the computation of D_k is

negligible while the computation of L_k and U_k consists in scaling the nonzero entries of L and U. Finally, by construction, $(S_k)_{ii} \in (0, 1]$, $i = 1, \ldots, n$, and this ensures that the conditioning of the matrices L_k and U_k is at least as good as the conditioning of L and U, respectively [4].

We underline that the above described updating strategy represents a generalization to the unsymmetric case of approaches proposed in [3, 4]. In fact, in [3, 4] updating techniques for diagonally modified symmetric positive definite systems have been developed.

A further strategy is presented in [2] where sparse approximations of the matrices L^{-1} and U^{-1} are employed. Such approximations are built computing an incomplete factorization for A_s^{-1} of the form

$$A_s^{-1} \approx W\, D^{-1} Z^T, \tag{22}$$

where D is a diagonal matrix, and W and Z are sparse unit upper triangular matrices. Note that W and Z are sparse approximations to U^{-1} and L^{-1}, respectively and

$$B_s = W\, D^{-1} Z^T \tag{23}$$

is an explicit preconditioner for A_s. A possibility to construct this sparse approximate inverse preconditioner is to use the incomplete generalized Gram-Schmidt orthogonalization process with respect to the bilinear form associated to A_s given in [10]. Alternatively, one can first compute an ILU factorization of A_s and then approximately invert L and U[11].

Exploiting factorization (22) and the relation $A_s \approx Z^{-T}\, DW^{-1}$, equality $A_k = A_s + (A_k - A_s)$ yields

$$A_k \approx Z^{-T} \left(D + Z^T (A_k - A_s) W \right) W^{-1}.$$

Then,

$$B_k^I = W \, \left(D + Z^T (A_k - A_s) W) \right)^{-1} Z^T$$

represents the explicit ideal preconditioner.

Since the mid-term $D + Z^T (A_k - A_s) W$ may be full, the above ideal preconditioner cannot be used in practice. Then, in [2] cheap approximations of $D + Z^T (A_k - A_s) W$ have been considered. In particular, $A_k - A_s$ is replaced by the band matrix $[A_k - A_s]_\delta$, and $Z^T [A_k - A_s]_\delta W$ by the band matrix $[Z^T [A_k - A_s]_\delta W]_\beta$, for some nonnegative β and δ. Therefore, the updated explicit preconditioner for A_k is

$$B_k^E = W \, \left(D + [Z^T [A_k - A_s]_\delta W]_\beta \right)^{-1} Z^T. \tag{24}$$

Similar ideas have been employed to build updating strategies for preconditioning sequence of linear systems arising in deblurring and denoising image restoration [7].

We further observe that in (24) as well as in (17) when $\beta = 0$, the middle factor in P_k^B and B_k^E is diagonal and the application of the preconditioner is straightforward; on the other hand if $\beta > 0$ the application of B_k^E and of P_k^B requires the solution of one banded linear system. In terms of computational cost, only small values of β and δ are viable and if β is nonzero, direct methods for banded systems are convenient. Finally, in both approaches the main diagonal and δ lower and upper diagonals of A_k are needed for building the preconditioner. In a matrix-free setting one possibility is to compute the nonzero entries of the matrix $[A_k]_\delta$ exploiting the separability of the matrix-vector operator, as described in Section 3. A less expensive approach is based on the observation that in both cases $[A_k]_\delta$ is needed to compute the product $[A_k]_\delta R$, where R is the sparse triangular matrix W in (24) or the triangular, band matrix $[U^{-1}]_\gamma$ in (17). Then, letting R_j be the j-th column of R and

$$\phi_j = [A_k]_\delta R_j,$$

we can observe that ϕ_j has at most $\delta + j$ nonzero entries in case $R = W$ and at most $\delta + \gamma$ nonzero entries in case $R = [U^{-1}]_\gamma$. These nonzero entries can be computed as follows. Note that

$$\phi_{ij} = \sum_{l \in L} (A_k)_{il} (R)_{lj}, \tag{25}$$

where $L = \{l : l \in [l_m, l_M], l_m = \max\{i - \delta, 0\}, l_M = \min\{i + \delta, n\}\}$. Then

$$\phi_{ij} = (A_k w)_i = \mathcal{F}_i(w), \qquad \text{with} \qquad w = (0, 0, 0, R_{l_m,j}, \ldots, , R_{l_M,j}, 0, 0, 0)^T.$$

Therefore, the computation of each nonzero entry of ϕ_j requires the evaluation of one component of the function $\mathcal{F}$.

4.3 Accuracy of the updated preconditioners

Intuitively, the effectiveness of the updated preconditioners described in the previous subsection depends on the accuracy of the seed preconditioner, on the magnitude of the discarded quantities in the approximation of $A_k - A_s$ and on the quality of the approximations to L^{-1} and/or U^{-1}.

The following theorem summarizes the above considerations and the theoretical results given in [2, 8, 24] regarding the preconditioners P_k^L, P_k^U, P_k^B and B_k^E. Similar considerations on preconditioner P_k^D will follow. Concerning the accuracy of P_k^L and P_k^U, in the following analysis we limit ourselves to consider the update P_k^L in (15). Obviously, analogous results apply to P_k^U in (14). Let us introduce the following functions

$$o_\gamma(A) = A - [A]_\gamma,$$

for any matrix A and $\gamma \geq 0$, and

$$o(A_k - A_s) = \begin{cases} triu(A_k - A_s, 1) & \text{if } P_k = P_k^L \\ \\ o_\delta(A_k - A_s) & \text{if } P_k = (B_k^E)^{-1} \text{or } P_k = P_k^B \end{cases}$$

Note that $o(A_k - A_s)$ represents the discarded quantity in the approximation of $A_k - A_s$.

Theorem 4.1 *Let P_k be either P_k^L or P_k^B or $(B_k^E)^{-1}$. Then,*

$$\|A_k - P_k\| \leq \|A_s - P_s\| + \bar{c}\|o(A_k - A_s)\| + e_k \tag{26}$$

where $\bar{c}$ and e_k take the following form. If $P_k = P_k^L$, then

$$\bar{c} = \|U\|, \qquad e_k = \|I - U\|\|A_k - A_s\|;$$

if $P_k = P_k^B$, then $\bar{c} = 1$ and

$$\begin{aligned} e_k \quad = \quad & (\|L\|\|U\|\|o_\gamma(L^{-1})\|\|o_\gamma(U^{-1})\| + \|L\|\|o_\gamma(L^{-1})\| + \|U\|\|o_\gamma(U^{-1})\|)\|[A_k - A_s]_\delta\| \\ & + \|L\|\|U\|\|o_\beta([L^{-1}]_\gamma [A_k - A_s]_\delta [U^{-1}]_\gamma)\| \end{aligned}$$

if $P_k = (B_k^E)^{-1}$, then

$$\bar{c} = 1, \qquad e_k = \|Z^{-1}\|\|W^{-1}\|\|o_\beta(Z^T [A_k - A_s]_\delta W)\|.$$

Proof. Note that

$$A_k - P_k = (A_k - A_s) + (A_s - P_s) + (P_s - P_k). \tag{27}$$

Let us consider the quantity $P_s - P_k$ for each preconditioner. Focusing on (15) we have

$$P_s - P_k = -tril(A_k - A_s)U = -(A_k - A_s)U + o(A_k - A_s)U$$

and the thesis trivially follows from (27). Concerning preconditioner (17),

$$\begin{aligned} P_s - P_k &= -L([L^{-1}]_\gamma\,[A_k - A_s]_\delta\,[U^{-1}]_\gamma)U + L(o_\beta([L^{-1}]_\gamma\,[A_k - A_s]_\delta\,[U^{-1}]_\gamma))U \\ &= -L((L^{-1} - o_\gamma(L^{-1}))\,[A_k - A_s]_\delta(U^{-1} - o_\gamma(U^{-1}))U \\ &\quad -L(o_\beta([L^{-1}]_\gamma\,[A_k - A_s]_\delta\,[U^{-1}]_\gamma))U \\ &= -[A_k - A_s]_\delta - L(o_\gamma(L^{-1})\,[A_k - A_s]_\delta o_\gamma(U^{-1}))U + (\,[A_k - A_s]_\delta o_\gamma(U^{-1}))U \\ &\quad +L(o_\gamma(L^{-1})\,[A_k - A_s]_\delta) - L(o_\beta([L^{-1}]_\gamma\,[A_k - A_s]_\delta\,[U^{-1}]_\gamma))U \end{aligned}$$

Then, the thesis follows from (27) and the definition of e_k.

Finally, let us consider the case $P_k = (B_k^E)^{-1}$. Since $P_s = Z^{-T}DW^{-1}$, from (24) it follows:

$$\begin{aligned} P_s - P_k &= -Z^{-T}(Z^T[A_k - A_s]_\delta W)W^{-1} + Z^{-T}o_\beta(Z^T[A_k - A_s]_\delta W)W^{-1} \\ &= -[A_k - A_s]_\delta + Z^{-T}o_\beta(Z^T[A_k - A_s]_\delta W)W^{-1} \end{aligned}$$

and (27) yields the thesis. □

The previous theorem enlightens that $\|A_k - P_k\|$ depends on the accuracy of the seed preconditioner, on the discarded quantity $o(A_k - A_s)$ and on the magnitude of e_k. Note that, considering preconditioner P_k^L, e_k is small whenever U is close to the identity matrix. On the other hand, in the approach employing P_k^B the magnitude of e_k depends on the magnitude of the discarded quantities in the approximation of L^{-1} and U^{-1} as well in the approximation of the matrix $[L^{-1}]_\gamma[A_k - A_s]_\delta[U^{-1}]_\gamma$. Finally, when preconditioner B_k^E is considered the more accurate is the approximation $[Z^T[A_k - A_s]_\delta W]_\beta$ to $Z^T[A_k - A_s]_\delta W$, the smaller is e_k.

With this result at hand, it is possible to show that the updated preconditioners have the potential to be more effective than reusing P_s (*frozen preconditioner*). This is shown in the next lemma, whose proof follows the lines of Lemma 2.1 in [24].

Lemma 4.2 *Let P_s, P_k, $\bar{c}$ and e_k be given as in Theorem 4.1. Assume that P_s satisfies*

$$\|A_s - P_s\| = \epsilon\|A_s\| < \|A_s - A_k\|, \tag{28}$$

for some positive ϵ. Then

$$\|A_k - P_k\| \leq \frac{\epsilon\|A_s\| + \bar{c}\|o(A_k - A_s)\| + e_k}{\|A_k - A_s\| - \epsilon\|A_s\|}\,\|A_k - P_s\|. \tag{29}$$

Proof. Condition (28) provides the following bound

$$\begin{aligned} \|A_s - A_k\| - \epsilon\|A_s\| &= \|A_s - A_k\| - \|A_s - P_s\| \\ &\leq \|A_k - P_s\|. \end{aligned}$$

This fact along with (26) yields

$$\begin{aligned} \|A_k - P_k\| &\leq \frac{\epsilon\|A_s\| + \bar{c}\|o(A_k - A_s)\| + e_k}{\|A_k - A_s\| - \epsilon\|A_s\|}\,(\|A_k - A_s\| - \epsilon\|A_s\|) \\ &\leq \frac{\epsilon\|A_s\| + \bar{c}\|o(A_k - A_s)\| + e_k}{\|A_k - A_s\| - \epsilon\|A_s\|}\,\|A_k - P_s\|, \end{aligned}$$

which completes the proof. □

Note that the coefficient of $\|A_k - P_s\|$ in (29) can be smaller than 1 provided that the discarded quantity $\|o(A_k - A_s)\|$ and e_k are small.

Let us now consider the preconditioner $P_k = P_k^D$. It is possible to prove that also in this case (26) holds with $\bar{c} = 1$ and

$$e_k = \hat{c}\|\Sigma_k\|, \tag{30}$$

with $\hat{c} = 2\|\mathit{off}(L)\|\|\mathit{off}(U)\| + 3(\|\mathit{off}(L)\| + \|\mathit{off}(U)\|)$. In fact we have $o(A_k - A_s) = \mathit{off}(A_k - A_s)$ and

$$\begin{aligned} \|A_k - P_k^D\| &\leq \|A_k - (LDU + \Sigma_k)\| + \|LDU + \Sigma_k - P_k^D\| \\ &\leq \|A_s - P_s\| + \|A_k - A_s - \Sigma_k\| + \|LDU + \Sigma_k - P_k^D\|. \end{aligned}$$

By [8, Theorem 3.1] it follows

$$\|LDU + \Sigma_k - P_k\| \leq e_k,$$

with e_k given in (30). Then, since $A_k - A_s - \Sigma_k = \mathit{off}(A_k - A_s)$ we get (26).

We conclude this section observing that if $\|\cdot\|$ denotes the 1 or the infinity norm, inequality (26) yields

$$\|A_k - P_k\| \leq \|A_s - P_s\| + C\|A_k - A_s\|, \tag{31}$$

where C is a strictly positive constant. This is due to the fact that by the properties of the 1 or the infinity norm, $\|[A]_\gamma\| \leq \|A\|$ and for any matrix A and nonnegative γ and $\|o(A_k - A_s)\| \leq \|A_k - A_s\|$. Then, $\|A_k - P_k\|$ is small whenever P_s is an accurate preconditioner for A_s and if A_k is close to A_s. In other words the updating procedures are expected to be effective in case of slowly varying sequences. This is likely to appear in sequences arising in the last iterations of a Newton-Krylov solver, as observed in Subsection 2.1. On the other hand, this observation suggests that the performance of an update preconditioner may deteriorate when A_k is not close to A_s and therefore it is advisable to combine the updating procedures with adaptive technique for refreshing the preconditioner and building a new reference matrix A_s and a new seed preconditioner (see Section 6).

5 Quasi-Newton updates

The main idea of the works [12, 13, 34, 35, 37] is to build a sequence of preconditioners by imposing the secant condition as typical of the implementation of quasi-Newton methods.

For sequences of the form (4) arising in the solution of nonlinear systems using a Newton-like method, we describe the Broyden-type rank-one updates proposed in [12] and its variant for the special case where the Jacobian is SPD [13], while regarding sequences where the matrices A_k are SPD, as those arising in Truncated Gauss-Newton method for nonlinear least-squares, we review a different approach based on L-BFGS updating described in [37].

5.1 Low rank updates for sequences arising in Newton-Krylov methods

The use of structured quasi-Newton formulae as preconditioners for sequences of the form (4) arising in Newton-Krylov solvers was firstly proposed in some pioneering papers by Martinez, see e.g. [34, 35].

In particular, Martinez introduced a new class of preconditioners that are obtained combining a classical incomplete factorization preconditioner for Θ_k' with the "structured least-change secant

update" (LCSU) framework. Then, in this approach a preconditioner P_k^{LCSU} for the kth system of the sequence (4) has the form

$$P_k^{LCSU} = C_k + E_k, \tag{32}$$

with C_k including partial information on Θ'_k and E_k is updated using LCSU techniques [19]. In order to use P_k^{LCSU} as preconditioner, its inversion must be inexpensive and one may consider C_k as an incomplete factorization of Θ'_k and E_k as a low-rank matrix such that P_k^{LCSU} solves a secant equation.

Following this approach, the computation of P_k^{LCSU} is performed assuming that a preconditioner of choice C_k is built at every Newton iteration and enriched with the matrix E_k that is obtained by low-rank update of E_{k-1}. The main focus of [34, 35] is on convergence analysis of a Newton-Krylov method incorporating the preconditioner explicitly in the definition of the trial point. In other words, a modification of the classical Newton-Krylov method is proposed where the trial step that solves $P_k^{LCSU} s = -\Theta_k$ is attempted and the convergence properties of the resulting method are studied, while the accuracy of the preconditioner is not analyzed.

On the other hand, Bergamaschi et al. proposed in [12], quasi-Newton updating techniques that belong to the class (32) focusing on the accuracy of the preconditioner at each nonlinear iteration and on the numerical implementation issues.

Given a seed matrix Θ'_s and either an explicit or an implicit seed preconditioner, the strategy proposed in [12] consists in updating this preconditioner using rank-one updates. Such an update is obtained imposing a secant equation that takes into account both the Newton step computed at the previous nonlinear iteration and the difference between the last two consecutive function values.

Let P_k be a preconditioner for Θ'_k, let s_k be the k-th Newton step and let $y_k = \Theta_{k+1} - \Theta_k$. Then the new updated preconditioner $P_{k+1} \approx \Theta'_{k+1}$ is obtained by imposing the secant condition

$$P_{k+1} s_k = y_k.$$

In the general unsymmetric case, P_{k+1} is uniquely determined by choosing the closest matrix to the current P_k satisfying the secant condition, i.e.

$$P_{k+1} = \underset{P:\ Ps_k = y_k}{\operatorname{argmin}} \|P - P_k\|_F$$

where $\|\cdot\|_F$ is the Frobenius norm. This yields the classical Broyden formula:

$$P_{k+1} = P_k + \frac{(y_k - P_k s_k) s_k^T}{s_k^T s_k} \tag{33}$$

and therefore P_{k+1} is obtained updating the previous preconditioner via Broyden-type rank-one updates.

Note that in this approach the computation of an incomplete factorization of the Jacobian at each Newton-Krylov iteration is not needed. In fact, preconditioner P_k belongs to the class (32) according to the following choice of the matrices C_k and E_k: $C_k = 0$ and $E_k = P_k$, except for the seed iteration $k = s$, where $C_s = P_s$ and $E_s = 0$.

Using the Sherman-Morrison formula the inverse B_{k+1} of P_{k+1} can be obtained by

$$B_{k+1} = B_k - \frac{(B_k y_k - s_k) s_k^T B_k}{s_k^T B_k s_k}. \tag{34}$$

In [12] the accuracy of the preconditioner in terms of the distance

$$\|I - P_k^{-1} \Theta'_k\|$$

is analyzed. The derivation of this bound differs in the classical analysis of Broyden methods as the trial step here is computed solving the Newton equation with the true Jacobian, while in Broyden method the trial step solves $P_k s = -\Theta_k$. The analysis is carried out under classical hypothesis on Θ for the convergence of Newton or Newton-like approaches and it is proved assuming that the seed Jacobian is Θ'_0. Then, if the initial guess x_0 is sufficiently close to the solution x^* of (2) and the initial seed preconditioner P_0 is sufficiently close both to Θ'_0 and to $\Theta'(x^*)$, then $\|I - P_k^{-1}\Theta'_k\|$ can be tuned to any fixed accuracy. In particular the following result is proved , see [12, Theorem 3.6].

Theorem 5.1 *Assume that (2) has a solution x^*, that $\Theta'(x)$ is Lipschitz continuous and $\Theta'(x^*)$ is nonsingular. Let $\alpha = \|\Theta'(x^*)^{-1}\|$. Then, fixed $\delta_1 \in (0, 1/\alpha)$ and $\hat{\delta}_1 > 0$, there exist $\delta, \delta_P, \hat{\delta}_P$ strictly positive such that if $\|x_0 - x^*\| \leq \delta$, $\|P_0 - \Theta'_0\| \leq \delta_P$ and $\|P_0 - \Theta'(x^*)\| \leq \hat{\delta}_P$, then*

$$\|I - P_k^{-1}\Theta'_k\| \leq \frac{\hat{\delta}_1 \alpha}{1 - \delta_1 \alpha} \qquad k = 1, 2, \ldots$$

In [13] the previous approach is specialized to sequences arising in nonlinear systems where the Jacobian are SPD matrices. In this case, the updating strategy is performed using BFGS rank-two updates so that the symmetry and the positive definiteness of the preconditioners is preserved. Then, given an SPD seed preconditioner Θ'_s, P_{k+1} and its inverse B_{k+1} are computed as follows

$$P_{k+1} = P_k + \frac{y_k^T y_k}{y_k^T s_k} - \frac{(P_k s_k)(P_k s_k)^T}{s_k^T P_k s_k},$$

and

$$B_{k+1} = \left(I - \frac{s_k^T y_k}{s_k^T y_k}\right) B_k \left(I - \frac{y_k^T s_k}{s_k^T y_k}\right) + \frac{s_k s_k^T}{s_k^T y_k}. \tag{35}$$

Note that if P_0 is SPD then P_k is SPD under the condition $s_k^T y_k > 0$ [31]. A result on the accuracy of the BFGS preconditioner (35) analogous to Theorem 5.1 was proved in [13, Theorem 3.6] with the further assumption that P_0 is SPD.

Interestingly, these ideas have been also used for preconditioning the sequence of linear systems arising in Newton methods for large symmetric eigenproblems [14].

The implementation of the above procedures can be made in a complete matrix-free manner without computing the preconditioner explicitly. In fact, assume to have a seed Jacobian $\Theta'_s \equiv \Theta'_{\bar{k}}$ and the corresponding seed preconditioner B_s, then the application of the preconditioner B_k in (34) and (35) can be performed using a recursive formula which require the computation of one matrix vector products with B_s, and a number of scalar products and daxpy operations that depends on $k - \bar{k}$.

Therefore, if B_s is in the form of an approximate inverse preconditioner, its application is straightforward and B_k results a "pure explicit preconditioner". Else if B_s is given as the inverse of an incomplete factorization, then $B_k \approx {\Theta'_k}^{-1}$ and its application requires the solution of two triangular sparse linear systems.

It should be underlined that quasi-Newton like preconditioners suffer from two main drawbacks, namely the increasing cost of memory for saving y_k and s_k and the increasing CPU time to apply the preconditioner. Moreover, the accuracy result given in Theorem 5.1 holds provided that x_0 is sufficiently close to x^*. Then, in practice restart procedures must be used especially if the number of nonlinear iterations is high. We postpone the discussion on this topic to Section 6.

As a final comment we note that these techniques are closely related to the Newton-Krylov setting, while the factorization updating approaches described in Section 4 can be applied to sequences arising in a more general context.

5.2 Limited-memory quasi-Newton updates for SPD sequences

In this section we consider the solution of sequences of systems (1) where the matrices A_k are SPD and assume to use the PCG method for approximately solving the linear systems. We describe a class of preconditioners, given in [37], built exploiting information collected by the CG method applied to the previous system of the sequence. This approach results in a limited-memory BFGS technique that can be applied to a general sequence of SPD linear systems, as those arising in the solution of nonlinear least-squares. In [37] special attention is devoted to the context of solution of large-scale unconstrained minimization problems where the matrices A_k corresponds to the Hessian of the objective function evaluated at the current point.

The basic idea is to observe that solving the SPD system $A_k s = b_k$ is equivalent to minimizing the convex quadratic function

$$q_k(s) = \frac{1}{2} s^T A_k s - b_k^T s$$

whose gradient is given by the residual $r(s) = A_k s - b_k$ of the linear system.

Then, if a BFGS formula is computed exploiting information collected by a minimization process for q_k, then an SPD approximation of the inverse of the Hessian of q_k, that is A_k, is obtained. This approximation can be used as an explicit preconditioner for the minimization of the subsequent system $A_{k+1} s = b_{k+1}$.

Let $\{s^i\}$ and $\{r^i\}$ denotes the sequence of iterates and residuals generated by the CG method applied to the solution of the linear system $A_k s = b_k$ and assume that the following m vector-pairs (w^i, y^i) have been computed and stored:

$$w^i = s^{i+1} - s^i, \quad y^i = r^{i+1} - r^i, \quad i = l_1, \ldots, l_m. \tag{36}$$

Then, the limited memory BFGS approximation to the inverse of the Hessian A_k of $q_k(s)$, built using the m vector-pairs (36), is given by [39]

$$\begin{aligned} H(m) \;=\; & (V_{l_m}^T \cdots V_{l_1}^T) \bar{H} (V_{l_1} \cdots V_{l_m}) \\ & + \rho_{l_1} (V_{l_m} \cdots V_{l_2}) w^{l_1} (w^{l_1})^T (V_{l_2} \cdots V_{l_m}) \\ & + \rho_{l_2} (V_{l_m} \cdots V_{l_3}) w^{l_2} (w^{l_2})^T (V_{l_3} \cdots V_{l_m}) \\ & \vdots \\ & + \rho_{l_m} w^{l_m} (w^{l_m})^T \end{aligned} \tag{37}$$

with

$$\rho_i = 1/(y^i)^T (w^i), \quad V_i = I - \rho_i y^i (w^i)^T, \quad i = l_1, \ldots, l_m,$$

and $\bar{H}$ to be

$$H_l = \frac{(w^l)^T y^l}{(y^l)^T y^l} I,$$

where l denotes the last correction pair generated in the previous CG cycle. Then $B_k = H(m)$ is used as an approximate inverse preconditioner for the next system of the sequence. The same scheme can be repeated for the subsequent systems of the sequence always using CG information of the mostly recently solved system. In this strategy the first system of the sequence is solved by the unpreconditioned CG method and the m vectors-pairs (36) are computed and stored.

The parameter m determines the amount of memory in the preconditioner and is normally chosen to be much smaller than the number of variables so that the cost of applying the preconditioner is not too large. There exist different strategies to choose the m correction pairs to be

used at each iteration. One considers simply the last m pairs, another saves vectors which are approximatively randomly distributed throughout the CG run.

The effectiveness of the automatic preconditioner (37) has been extensively investigated in [37] including experiments within a Hessian-free Newton method for solving unconstrained optimization problem and tested on different strategies for choosing the m pairs (36). Results seem to indicate that the limited memory BFGS preconditioner may be useful when the coefficient matrices A_k are not very sparse or when A_k is not explicitly available and products of A_k times vectors are expensive to compute.

6 Refresh and restart strategies

Generally, the updating procedures presented in this survey require the use of restarting techniques to refresh the seed preconditioner.

In particular, the effectiveness of preconditioners presented in Section 4 depends on the difference between A_k and A_s (see (31)) and when this difference becomes large, the quality of the updated preconditioner P_k can deteriorate and the convergence of the linear solver can slow down. To handle this issue, in the proposals [2, 8] the quality of the preconditioner is checked out. More precisely, if the linear solver fails in computing an inexact Newton step with the prescribed accuracy a new reference matrix and a new preconditioner are initialized. In [15] periodic re-factorizations of the seed preconditioner are used. Moreover, it is also noted that in CFD applications parts of the sequence of linear systems show large entries in the difference matrices and in other parts system matrices are very close. In the latter case the use of the updating procedure seems to be unproductive as the frozen preconditioner is powerful. Then, in [15] an adaptive technique to avoid unnecessary updating is adopted.

On the other hand, algorithms based on quasi-Newton formula typically suffer from the increasing cost of memory for saving the work vectors y_k and s_k. Moreover, if the number of linear systems to be solved is high (e.g. more than ten nonlinear iterations are performed), the application of the preconditioner may be too heavy to be counterbalanced by a reduction in the iterations number of the iterative linear solver. In [37] the memory cost is implicitly kept low by using the limited memory implementation. The option followed in [12, 13] consists in recomputing the seed preconditioner after a fixed number k_{max} of iterations, see Section 7. The fine tuning of k_{max} is not a trivial task and it has been addressed in [13]. The numerical experiments performed there suggest that k_{max} belonging to the interval $[1, 5]$ might be a good choice to get a sufficiently accurate preconditioner and to keep the overall overhead of the algorithm low.

7 Numerical illustration

In this section we give a numerical illustration of the typical behaviour of updated preconditioners presented in this survey. We focus on sequences arising in Newton-Krylov methods for nonlinear systems of the form (4) and we selected two preconditioning strategies. One is the strategy based on the updating of the factorized approximate inverse seed preconditioner B_k^E given in (24) (FINVUP strategy); the other is the Broyden-type updating given in (34) (BroyUP strategy). The reported results are illustrative of the behaviour of the updating strategies falling in the two classes reviewed in this survey. We compare the behaviour of these updating procedures with those of the frozen preconditioner (Freeze strategy) and the recomputed preconditioner (Recomp strategy). In the Freeze strategy the preconditioner is computed only for the first reference matrix Θ'_0 and reused in all the subsequent nonlinear iterations; in the Recomp strategy a preconditioner for each linear system of the sequence is recomputed from scratch.

BiCGSTAB is used as Krylov solver and the linear inner iterations are stopped when condition (5) is met. We refer the reader to [2, 12] for details on the implementation of the Newton-Krylov method.

Concerning the FINVUP preconditioner, we summarize the numerical experience given in [2]. The seed Jacobian used was the first reference matrix of the sequence, and an approximate sparse inverse preconditioner B_s is constructed using an incomplete LDU factorization with drop tolerance of the seed matrix followed by the approximate inversion of the factors L and U. A drop tolerance has been used to perform the approximate inversion of the triangular factors.

Four problems widely used in literature were considered: the Nonlinear Convection-Diffusion problem (`NCD`), the Flow in a Porous Medium problem (`FPM`), the CounterCurrent Reactor problem (`CCR`) and the 2D Driven Cavity problem (`2DC`). Three of them, namely problems `NCD`, `FPM` and `2DC`, arise from the discretization of PDE problems. In all four problems, the dimension n of the nonlinear systems was varied considering values ranging from 6400 to 62500. `NCD` has been solved setting the Reynolds number equal to 250, 500, 1000 while `2DC` has been solved with Reynolds numbers 200, 250, 300 and 350. All these settings gave rise to a testing set made of 22 problems.

The update of the preconditioner was performed allowing for a diagonal approximation ($\delta = 0$ and $\beta = 0$) in (24) when sequences arising in the solution of problems (`FPM`) and (`CCR`) are solved, while tridiagonal banded approximation ($\delta = 1$ and $\beta = 1$) are used in the solution of sequences arising in (`FPM`) and (`2DC`) problems.

To compare the overall computational effort of the three preconditioning strategies (Freeze, Recomp, FINVUP), the performance profiles proposed by Dolan and Moré [23] was used. For each problem P in the testing set and each Algorithm A, let $\texttt{LI}_{P,A}$ denote the total number of linear iterations required to solve problem P using Algorithm A and $\texttt{LI}_P$ be number of linear iterations required by the best algorithm to solve problem P, i.e. the algorithm which uses the fewest linear iterations. Then, the linear iterations performance profile is defined for the algorithm A as

$$\pi_A(\tau) = \frac{\text{number of problems s.t. } \texttt{LI}_{P,A} \leq \tau\, \texttt{LI}_P}{\text{number of problems}}, \quad \tau \geq 1.$$

Analogously, we define the CPU time performance profile $\phi_A(\tau),\ \tau \geq 1$ measuring the efficiency of the algorithm A in terms of the employed CPU time.

Here we use two different quantities to measure the computational effort of each strategy: the number of BiCGSTAB iterations performed and the execution time needed to solve each test (see Figure 2).

The performance profiles show the typical behaviour of a factorization updating strategy: the Recomp strategy is the most effective in terms of linear iterations, while FINVUP outperforms the Freeze approach. The situation is quite different when we consider execution time. The repeated computation of the factorization of the preconditioner is relatively expensive and therefore the recomputation strategy is in general less time efficient than updating. In fact, FINVUP is the most efficient in the solution of 73% of the tests. The remaining tests successfully solved by FINVUP require a computational effort that is at most two times the effort required by the best solver. Note also that FINVUP is the most reliable strategy as it solved all the tests.

In Figure 3 we focus on the sequence arising in the solution of the `NCD` with $n = 22500$ and $Re = 500$. We plot, for each nonlinear iteration, the number of linear iterations performed by BiCGSTAB coupled with the Frozen and with the FINVUP strategy. The figure clearly shows that the performance of the frozen preconditioner deteriorates soon during the solution of the sequence and is rather unpredictable, while the updating strategy deteriorates only in the last two iterations. Note the seed preconditioner has never been recomputed.

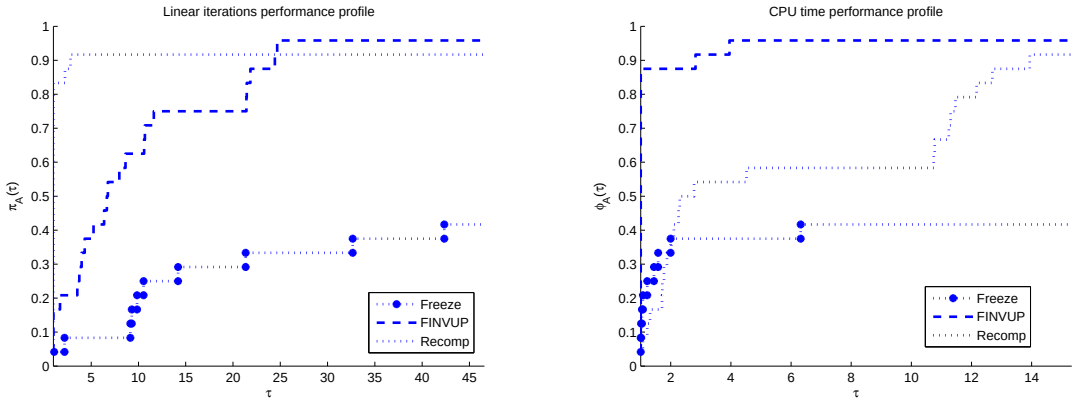

Figure 2: Performance profile in terms of linear iterations (left) and execution time (right)

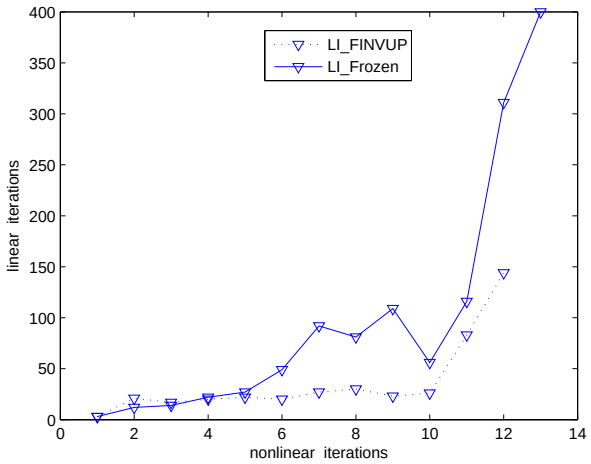

Figure 3: Problem NCD with $n = 22500$ and $Re = 500$: comparison, in terms of linear iterations between the Frozen and the FINVUP strategy

	$B_s = ILU(0)$				$B_s^{-1} = AINV(0.1)$			
B_k	k_{max}	NLI	LI	CPU	k_{max}	NLI	LI	CPU
Freeze	-	7	851	11.59	-	7	882	18.35
Recomp	-	7	754	8.65	-	7	908	19.7
BroyUP	1	7	442	6.51	1	8	574	14.62
BroyUP	2	7	470	6.56	2	7	517	13.07
BroyUP	3	7	501	6.93	3	7	647	16.1
BroyUP	5	7	529	7.09	5	7	561	14.08
BroyUP	∞	6	515	9.67	∞	7	655	17.15

Table 1: Results of BroyUP preconditioner on the Bratu problem.

Next we focus on the Broyden-type update (34) and we show some of the results obtained in [12]. Specifically, we consider the sequence arising in the solution of the classical Bratu problem. In [12] the problem has been discretized by 2D Mixed Finite Elements yielding to a sequence of systems with $n = 28600$. Concerning the employed restart strategy, this is ruled by the choice of the parameter k_{max} which represents the the maximum number of rank one corrections allowed. Each k_{max} nonlinear iterations, the last k_{max} computed vectors $s_j, y_j, j = k - k_{max} + 1, \ldots, k$ needed to compute the current preconditioner are replaced with the last computed s_k, y_k and a new seed preconditioner B_s is formed.

In Table 1 we show the value of k_{max}, the number of nonlinear iterations (NLI), the number of BiCGSTAB iterations (LI), the total CPU time in seconds. The first columns of the table refer to tests where the ILU(0) factorization is used to compute the seed preconditioner B_s and the last columns refer to the case where the AINV factorization with drop tolerance 0.1 is used for computing B_s^{-1}. Small values of the allowed quasi-Newton corrections k_{max} are reported and compared with the choice $k_{max} = \infty$ which corresponds to not employ the restarting procedure and with the Frozen and Recomp strategy.

With both seed preconditioners (ILU(0) or AINV(0.1)) the use of BroyUP produces an acceleration in terms of both number of iterations and CPU time. Moreover, the restarting strategy enhances the performance of the overall algorithm since the choice $k_{max} = \infty$ yields to the less efficient implementation. This is not surprising since the results of Theorem 5.1 only hold when x_0 is near the solution. With the restarting strategy, the seed preconditioner is computed every k_{max} iterations, thus taking advance from the nonlinear convergence. We note that these updating strategies outperform the Recomp strategy also in terms of linear iterations. This seems to be due to the fact that, as opposite to the preconditioner updating factorizations, the preconditoner is periodically recomputed and enriched by Broyden-rank one information on the current Jacobian.

8 Conclusion

The numerical solution of the linear algebra phase is crucial for effectiveness of optimization methods and often dominates the computational time, especially when large-scale problems are considered. Thus, the effectiveness of optimization algorithms is highly dependent on reliable and effective tools for numerical linear algebra. In this paper we have attempted to highlight some of the developments that have taken place in recent years in preconditioning techniques for sequences of linear systems arising in optimization methods. In particular we focused on preconditioner updating techniques for sequences arising in the solution of nonlinear systems and

nonlinear least-squares problems by Newton-Krylov methods. We described two main classes of updating preconditioners, that is the class based on updating available factorizations and the class based on quasi-Newton formula. We provided a uniform analysis of their accuracy showing their advantages and their drawbacks. Finally, the implementation of both classes has been discussed in a matrix-free setting and an overview of the practical behaviour of these strategies has been reported.

References

[1] S. Bellavia, S. Berrone, *Globalization strategies for Newton-Krylov methods for stabilized FEM discretization of Navier-Stokes equations*, Journal of Computational Physics, 226 (2007), pp. 2317-2340.

[2] S. Bellavia, D. Bertaccini, B. Morini, *Nonsymmetric preconditioner updates in Newton-Krylov methods for nonlinear systems*, SIAM J. Sci. Comput., 33 (2011), pp. 2595-2619.

[3] S. Bellavia, V. De Simone, D. di Serafino, B. Morini, *Efficient preconditioner updates for shifted linear systems*, SIAM J. Sci. Comput., 33 (2011), pp. 1785–1809.

[4] S. Bellavia, V. De Simone, D. di Serafino, B. Morini, *A preconditioning framework for sequences of diagonally modified linear systems arising in optimization*, SIAM J. Numer. Anal., 50 (2012), pp. 3280–3302.

[5] S. Bellavia, B. Morini, *A globally convergent Newton-GMRES subspace method for systems of nonlinear equations*, SIAM J. Sci. Comput., 23 (2001), pp. 940–960.

[6] S. Bellavia, B. Morini, *Subspace trust-region methods for large bound-constrained nonlinear equations*, SIAM Journal on Numerical Analysis 44, pp. 1535-1555, 2006.

[7] D. Bertaccini, F. Sgallari, *Updating preconditioners for nonlinear deblurring and denoising image restoration*, Applied Numerical Mathematics, 60 (2003), pp. 994-1006.

[8] S. Bellavia, B. Morini, M. Porcelli *New updates of incomplete LU factorizations and applications to large nonlinear systems*, Optimization Methods and Software, 29:2 (2014), pp. 321–340.

[9] M. Benzi, G.H. Golub, *Bounds of the entries of matrix functions with applications to preconditioning*, BIT, 39 (1999), pp. 417-438.

[10] M. Benzi, M. Tůma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 968–994.

[11] M. Benzi, M. Tůma, *A Comparative Study of Sparse Approximate Inverse Preconditioners*, Appl. Numer. Math., 30 (1999), 305–340.

[12] L. Bergamaschi, R. Bru, A. Martínez, M. Putti, *Quasi-Newton preconditioners for the inexact Newton method*, Electronic Trans. Num. Anal., 23 (2006) pp. 76–87.

[13] L. Bergamaschi, R. Bru, A. Martínez, *Low-rank update of preconditioners for the inexact Newton method with SPD Jacobian*, Mathematical and Computer Modelling, 54 (2011), pp. 1863–1873.

[14] L. Bergamaschi, A. Martínez, *Parallel RFSAI-BFGS preconditioners for large symmetric eigenproblems*, Journal of Applied Mathematics, 2013(2013), (10 pages).

[15] P. Birken, J. Duintjer Tebbens, A. Meister, M. Tůma, *Preconditioner updates applied to CFD model problems*, Appl. Numer. Math., 58 (2008), pp. 1628–1641.

[16] P.N. Brown, *A local convergence theory for combined inexact-Newton/finite-difference projection methods*, SIAM J. Numer. Anal., 24 (1987), pp. 407–434.

[17] P. N. Brown, Y. Saad, *Convergence theory of nonlinear Newton–Krylov algorithms*, SIAM J. Optim., 4 (1994), pp. 297–330.

[18] C. Calgaro, J.P. Chehab, Y.Saad, *Incremental incomplete ILU factorizations with applications*, Numerical Linear Algebra with Applications, 17 (2010), pp. 811–837.

[19] J. E. Dennis, R. B. Schnabel, *Unconstrained Optimization and Nonlinear Equations*, SIAM, 1983.

[20] M. D'Apuzzo, V. De Simone, D. di Serafino, *On mutual impact of numerical linear algebra and large-scale optimization with focus on interior point methods*, Computational Optimization and Applications, 45 (2010), pp. 283–310.

[21] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 400–408.

[22] S. Demko, W.F. Moss, P.W. Smith, *Decay rates for inverses of band matrices*, Mathematics of Computation, 43 (1984), pp. 491–499.

[23] E.D. Dolan, J.J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 91 (2002), pp. 201–213.

[24] J. Duintjer Tebbens, M. Tůma, *Efficient Preconditioning of Sequences of Nonsymmetric Linear Systems*, SIAM J. Sci. Comput., 29 (2007), pp. 1918–1941.

[25] J. Duintjer Tebbens, M. Tůma, *Preconditioner Updates for Solving Sequences of Linear Systems in Matrix-free Environment*, Numer. Linear Algebra Appl., 17 (2010), pp. 997–1019.

[26] S.C. Eisenstat, H.F. Walker, *Globally convergent inexact Newton methods*, SIAM J. Optim., 4 (1994), pp. 393–422.

[27] G. Fasano, M. Roma, *Preconditioning Newton–Krylov Methods in Non-Convex Large Scale Optimization*, Computational Optimization and Applications, 56, (2013), pp. 253–290.

[28] G. Fasano, F. Lampariello, M. Sciandrone, *A Truncated Nonmonotone Gauss-Newton Method for Large-Scale Nonlinear Least-Squares Problems*, Computational Optimization and Applications, 34, (2006), pp. 343–358.

[29] S. Gratton, A. S. Lawless, N. K. Nichols, *Approximate Gauss-Newton Methods for Nonlinear Least Squares Problems*, SIAM J. Optim., 18, (2007) pp. 106–132.

[30] M.R. Hestenes, *Pseudoinverses and conjugate gradients*, Communications of the ACM, 18 (1975), pp. 40–43.

[31] C.T. Kelley, *Iterative Methods for Optimization*, SIAM, 1995.

[32] D.A. Knoll, D.E. Keyes, *Jacobian-free Newton-Krylov methods, a survey of approaches and applications*, J. Comput. Phys., 193 (2004), pp. 357-397.

[33] D. Loghin, D. Ruiz, A. Touhami, *Adaptive preconditioners for nonlinear systems of equations*, Journal of Computational and Applied Mathematics, 189, (2006), pp. 362 - 374.

[34] J.M. Martínez, *A theory of secant preconditioners*, Math. Comp. 60 (1993) 681-698.

[35] J.M. Martínez, *An extension of the theory of secant preconditioners*, in: Linear/Nonlinear Iterative Methods and Verification of Solution, Matsuyama, 1993, J. Comput. Appl. Math. 60 (1995) 115–125.

[36] G. Meurant, *A review on the inverse of symmetric tridiagonal and block tridiagonal matrices*, SIAM J. Matrix. Anal. Appl., 13 (1992), pp. 707-728.

[37] J.L. Morales, J. Nocedal, *Automatic preconditioning by limited memory quasi-Newton updating*, SIAM J. Optim. 10 (2000) 1079–1096.

[38] R. Nabben, *Decay rates of the inverse of nonsymmetric tridiagonal and band matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 820–837.

[39] J. Nocedal, S.J. Wright, *Numerical Optimization*, Springer Ser. Oper. Res., Springer-Verlag, New York, 1999.

[40] C. C. Paige, M. A. Saunders,*LSQR: An algorithm for sparse linear equations and sparse least squares*, TOMS, 8, (1982), 43–71.

[41] M.L. Parks, E. de Sturler, G. Mackey, D.D. Johnson, S. Maiti, *Recycling Krylov subspaces for sequences of linear systems*, SIAM J. Sci. Comput., 28 (2006), pp. 1651–1674.

[42] M. Porcelli, *On the convergence of an inexact Gauss-Newton trust-region method for nonlinear least-squares problems with simple bounds*, Optimization Letters, 7:3 (2013), pp. 447–465.

[43] W. Xu, T. F. Coleman, *Solving Nonlinear Equations with the Newton-Kyrlov Method Based on Automatic Differentiation*, Journal of Optimization Methods and Software, 29(1) (2014), pp. 88–101.

[44] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd edition, SIAM, 2003.